

Attorney Docket No. IBM/161
Confirmation No. 9188

#27
S. Sand
9/26/02
PATENT

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte Randall R. Schnier

Appeal No. _____
Application No. 08/818,185

APPEAL BRIEF

RECEIVED

SEP 25 2002

Technology Center 2100

RECEIVED
SEP 25 2002 PM 1:40
BOARD OF PATENT APPEALS
AND INTERFERENCES

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Randall R. Schnier Art Unit: 2152
Serial No.: 08/818,185 Examiner: Thong Vu
Filed: March 14, 1997 Atty. Docket No.: IBM/161
For: A BOOTSTRAPPING TECHNIQUE FOR DISTRIBUTED OBJECT CLIENT
SYSTEMS

Assistant Commissioner for Patents
ATTENTION: Board of Patent Appeals and Interferences
Washington, D.C. 20231

APPEAL BRIEF

I. REAL PARTY IN INTEREST

This application is assigned to International Business Machines Corporation, of Armonk, New York.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-9, 11-15-36 and 39 are pending in the Application, with claims 1-3, 7, 11, 18-19, 21, 24-25, 28, 34, and 39 being once amended, claims 15-17 being twice amended, and claims 10, 12-14, and 37-38 being canceled. All pending claims stand rejected, and are now on appeal.

IV. STATUS OF AMENDMENTS

There have been no amendments filed subsequent to the last rejection (Paper 23).

V. SUMMARY OF INVENTION

Applicant's invention is generally directed to improvements in the area of "bootstrapping" a zero install client in a distributed object client-server computer system. The improvements principally are obtained by way of the automated download of an object reference to a naming context object from a server to a client in response to a request from the client after program code for a zero install client has been downloaded to a client machine.

As discussed, for example, starting at page 3, line 3 of the application, a distributed object system facilitates the interaction between objects in an object-oriented environment when objects are distributed among different machines coupled over a communications link. As is well known in the art, an object-oriented environment is one in which logical entities in a computer program are organized as "objects" that encapsulate both data and operations that may be performed on such data.

A distributed object system permits client and server computer systems to interact with one another through interaction between the client objects resident on client systems and server objects resident on server systems. To facilitate such interaction, a standardized protocol known as the Common Object Request Broker Architecture (CORBA) standard has been developed. As discussed at page 10, line 2 of the application, the CORBA standard facilitates interoperability between clients and servers on various platforms to communicate via Object Request Brokers (ORBs) defined in a language-independent specification known as the Interface Definition Language (IDL).

Using the CORBA standard, an object resident on a server computer can be accessed on a client computer through the installation of a proxy object on the client computer that, from the perspective of a client application that accesses the object, emulates the original object on the server. In object-oriented terminology, the proxy object is said to share the same "interface" as the original object, meaning that the methods capable of being invoked on the original object, and the format of the data returned as a result of the execution of those methods, is the same on both the proxy object and the original object. As a result, a client application configured to access the original object can also access the proxy object with no modification to its code.

A proxy object, however, internally differs from its original object in that functionally the proxy object does nothing more than pass along method requests through an ORB and onto the original object. Thus, through the interaction of a proxy object and its original object through an ORB, the fact that the original object is resident on a server computer is effectively hidden from the client application.

One basic concept of object-oriented programming that should also be understood in connection with this application is the difference between an "object" and a "class." As discussed, for example, at page 9 of the application, a "class" is a template that defines a type of object, including the methods and data encapsulated within a class of objects. Objects themselves are considered to be "instances" of a particular class. Multiple objects can be instantiated from a given class, however, so it is often necessary for a client to be able to distinguish from among multiple instantiations of an object class.

The example discussed starting at line 21 of page 11 of the application highlights this distinction through the discussion of a "checking account" object and class. A checking account class might include methods such as "deposit," "withdrawal," and "get_balance," such that any objects created from that class would include those methods. Given that a bank may have checking accounts for a multitude of individuals, multiple objects of the checking account class could be created, e.g., for customers such as John, Susan, etc. When it is desirable to perform an operation on John's checking account, therefore, a client application is typically required to interact with the checking account object for John, rather than on the class as a whole.

Under the CORBA standard, each original object on a server that is accessed by a client application is required to have a corresponding proxy object resident on the client. However, before a client can obtain a proxy object, the client is required to obtain an Object Reference (OR) for the proxy object so that the correct proxy object may be retrieved from the server. As described at page 10, line 22 of the application, an Object Reference is a set of data that uniquely identifies an object so as to distinguish that object from other objects. Once an Object Reference for a proxy object is obtained by a client, the actual proxy object can be retrieved and associated with the original object through a process known as "binding." Once bound, a proxy object has a logical pathway to its corresponding original object such that methods can be invoked on the

original object via calls to the proxy object, and results returned to the client via the proxy object through the CORBA architecture.

To support the retrieval of proxy objects, CORBA defines an entity known as a Naming Context Object (NCO) that is capable of obtaining and returning proxy objects for server objects requested by a client.

As discussed in the application starting at page 13, given that the NCO is also a server object, before a client program is able to call a binding method on a Naming Context Object, a proxy for the NCO itself must first be obtained by the client.

The process of dynamically creating a proxy object for an NCO is generally referred to as "bootstrapping." Specifically, bootstrapping involves obtaining a proxy for an NCO on a client machine, so that the NCO can be used to obtain proxies for other objects with which the client wishes to interact.

CORBA supports a method called "resolve_initial_references" which causes the ORB to request a proxy for an NCO. However, for an ORB to be able to request a proxy for the NCO, it must first have an Object Reference for the NCO, which is typically required to be in the form of a stringified Object Reference (OR).

With this background, Applicant's invention attempts to address the bootstrapping problem in a special circumstance, that of a "zero install" client.

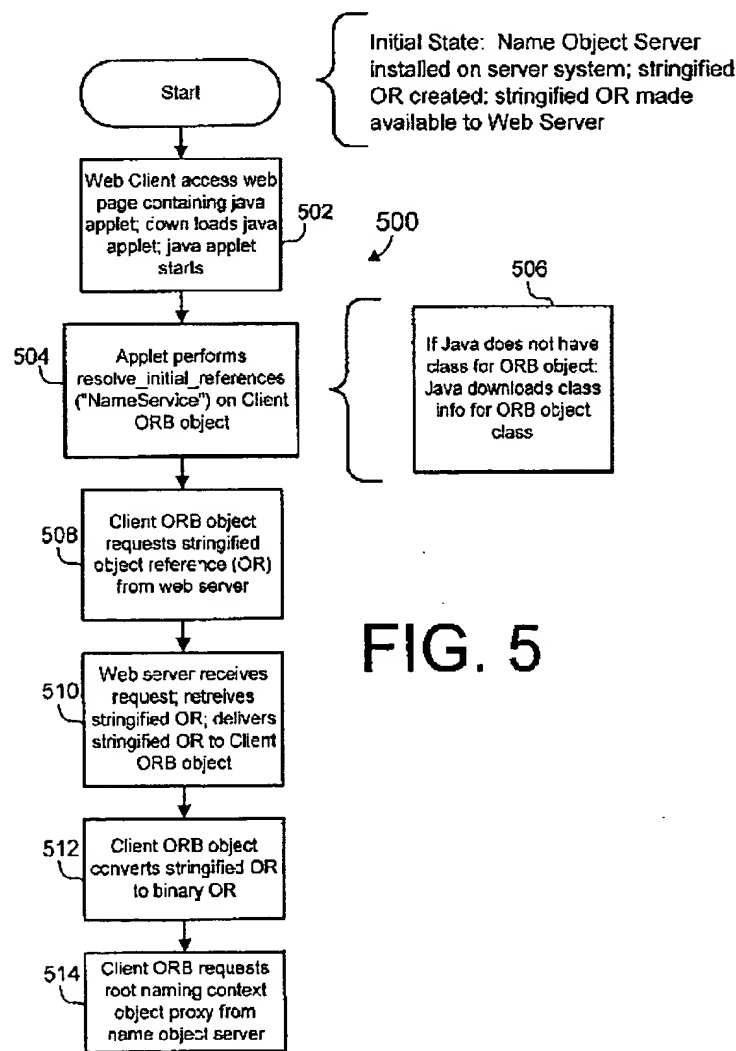
A zero install client is defined in the Application, beginning at page 3, line 15, to be an environment where no traditional client installation, including the installation of configuration information, is performed on the client. The predominant example of a zero install client is a web browser configured to download and execute platform independent program code such as Java applets. A virtual machine in a web browser may be used to execute Java or other platform independent code dynamically after the code has been downloaded to a web browser, and without any traditional "installation" of the program code on the client machine.

In addition to applets, Java has also been used to implement the various objects utilized in a CORBA environment, including the ORB client program code that is utilized to interface proxy objects with original objects residing on a server. However, as discussed at page 14, line 8 of the application, when both an applet and an ORB are implemented in Java, neither requires any form

of traditional installation. Moreover, as discussed beginning at line 13 of page 14, the lack of traditional installation prevents the stringified OR for an NCO to be made available to the client system upon download of the Java applet or ORB client to the client system.

Applicant's invention addresses this bootstrapping problem by having the object reference for the Naming Context Object delivered to a zero install client in response to a request to a server by the zero install client (Application page 4, lines 14-17, and Fig. 5 and accompanying disclosures).

One exemplary process for bootstrapping a zero install client in a manner consistent with the invention is illustrated in Fig. 5 of the application, which is reproduced below for the Board's convenience:



As shown in this figure, this embodiment of the invention operates by downloading a zero install client in the form of a Java applet (block 502), and optionally the class for an ORB object (block 506). The applet initiates the retrieval of an Object Reference to a Naming Context Object by performing a "resolve_initial_references ("NameService")" on the client ORB object (block 504), and consistent with the invention, the client ORB object requests a stringified OR for the Naming Context Object from an external web server (block 508). This request is processed by the web server to deliver the requested object reference to the client ORB object, which is converted to a binary OR and used by the client ORB to request the root NCO proxy object from a Name Object server (block 510-514).

An important aspect of this operation is that it is via the applet or web browser, a zero install client, that a request is made for the OR of the NCO. Moreover, this request is made after the applet is downloaded and executed by the web browser on a client machine. Thus, Applicant's claims are directed to a precise manner of bootstrapping a zero install client through a dynamic request made by the zero install client to retrieve an OR for an NCO.

VI. ISSUES

- A. Whether claims 1-9, 11, and 15-32 were improperly rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,009,466 to Hamilton et al. (hereinafter "*Hamilton*") in view of "*JavaOne, Remote object for Java*," Kessler et al., JAVAONE '96 Presentations, www.javasoft.com/javaone/javaone96/pres, (hereinafter "*Kessler*").
- B. Whether claims 33-36 and 39 were improperly rejected under 35 U.S.C. § 103(a) as being unpatentable over *Hamilton* in view of *Kessler* and further in view of U.S. Patent No. 5, 793,365 to Tang et al. (hereinafter, "*Tang*").

VII. GROUPING OF CLAIMS

For the purposes of appeal, the following groupings of claims are considered to be separately patentable, with the individual claims within each claim grouping standing or falling together:

Group I: claims 1-9, 11 and 13-32

Group II: claims 33-36 and 39

VIII. ARGUMENT

Applicant respectfully submits that the Examiner's obviousness rejections of claims 1-9, 11, 15-36 and 39 are not supported on the record, and that the rejections should be reversed. A *prima facie* showing of obviousness requires that the Examiner establish that the differences between a claimed invention and the prior art "are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art." 35 U.S.C. §103(a). Such a showing requires that all claimed features be disclosed or suggested by the prior art. Such a showing also requires objective evidence of the suggestion, teaching or motivation to combine or modify prior art references, as "[c]ombining prior art references without evidence of such a suggestion, teaching or motivation simply takes the inventor's disclosure as a blueprint for piecing together the prior art to defeat patentability -- the essence of hindsight." In re Dembiczak, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999).

Applicant respectfully submits that, in the instant case, the Examiner has failed to establish a *prima facie* case of obviousness as to any of the pending claims, and as such, the rejections should be reversed. Specific discussions of the non-obviousness of each of the aforementioned groups of claims are presented hereinafter.

A. The Group I Claims (claims 1-9, 11 and 13-32) were improperly rejected under 35 U.S.C. § 103(a) as being unpatentable over *Hamilton* in view of *Kessler*.

Claim 1, which is representative of the Group I claims, recites in part a computer program enabling client object-server object interaction for a client object located on a zero install client by delivering an object reference for a Naming Context Object to the zero install client after the zero install client has contacted the computer program. The other Group I independent claims (claims 7, 21, and 28) vary in scope from one another, as well as from claim 1, but all commonly recite the basic concept of delivering an Object Reference for a Naming Context Object to a client after that client has contacted or requested the Object Reference from a server.

Applicant respectfully submits that this feature is not disclosed or suggested by the combination of *Hamilton* and *Kessler*, nor by any of the other references cited by the Examiner.

As an initial matter, Applicant wishes to note that *Hamilton* and *Kessler* disclose essentially the same subject matter as a number of other prior art references utilized by the Examiner in previous Office Actions, including, among others, two references to *Phillips* and the *Orchard* reference entitled "Java and Objects." Over the course of the past two years, the Examiner has consistently applied new art against the claims, with none of the art providing any additional relevant disclosure over earlier-cited art. The Board will also note that this application was appealed after receipt of a non-final Office Action, which was the fourth non-final Office Action the Examiner has issued since the claims at issue were first presented in their current form. In view of the Examiner's continued citation of new references disclosing essentially the same subject matter and repeated issuances of non-final Actions, Applicant has become of the opinion that a resolution of the patentability of this application by the Board is now appropriate.

Now to turning to the substance of the Examiner's rejections, the Examiner relies on *Hamilton* and *Kessler* for allegedly disclosing Applicant's claimed invention. In particular, the Examiner argues that *Hamilton* discloses delivering an Object Reference for a Naming Context Object to a zero install client after the zero install client has contacted a server. The Examiner relies on *Hamilton*, at column 4, lines 50-67, column 6, line 65 to column 7, line 24, column 7, lines 38-59 and column 9, lines 55-65 to support this conclusion. Moreover, the Examiner

apparently takes official notice that "the Naming Context Object is located on the object name server and [sic] well-known in the art," relying on *Cheng et al.* and *Phillips et al.* (November 9, 2001 Office Action, paragraph 4).

However, as Applicant has argued to the Examiner a number of times, including in the Response dated August 14, 2001, as well as the Amendment and Response filed September 18, 2000, the Examiner seems to be relying on disclosure in the cited references pertaining to the use of a Naming Context Object for retrieving Object References for other objects to be utilized by a client. Put another way, the Examiner is focusing on the language in these references related to obtaining Object References from a Naming Context Object, rather than obtaining an Object Reference for the Naming Context Object itself, which is the subject of the Group I claims.

The cited references all discuss, at the most, how a Naming Context Object is used to obtain Object References for other objects. Indeed, this is the well known purpose of a Naming Context Object. What the references do not disclose, and what is an important aspect of the Group I claims, is precisely how an Object Reference for the Naming Context Object (given that the NCO is also an object) is obtained for the purpose of retrieving a proxy object for the NCO, which facilitates for further usage of the NCO in a standard CORBA environment.

None of the passages in *Hamilton* cited by the Examiner disclose delivering an Object Reference for a Naming Context Object to a zero install client after the zero install client has contacted a server, contrary to the Examiner's assertion.

First, the passage at column 4, lines 50-67 of *Hamilton* discloses how stubs are used to interface a client application with an ORB client for interfacing with an actual object resident on a server. In this context, a "stub" is synonymous with a "proxy object." The passage is otherwise silent as to any Naming Context Object or Object Reference therefor.

Second, the passage at column 6, line 65 to column 7, line 24 of *Hamilton* discloses the use of a network name server which is disclosed to return a "machine address for a network server in response to an inputted network server name." The passage also discloses an object name server that returns "references to network servers in response to an inputted object name." It should be appreciated that a "network server" as defined in *Hamilton* is any application on a network that includes software object capable of being accessed remotely (column 1, lines 27-

30), and as such, a "network server" generally refers to a server application in a client-server system.

Hamilton does recognize that an application program on a client may not be able to initially communicate with a network server, but discloses that the application program may be given the logical name for the network server, and may refer to the network name server to find the network server machine address (column 7, line 10-14). *Hamilton* goes on to describe that once the application has located the network server machine, the application "refers to object name server 560 to find a reference to network server 570." (Column 7, lines 15-18).

Put in the language used by Applicant, this quoted passage refers to the operation of accessing a name service to obtain a reference to an application, or more likely, an Object Reference for an object encapsulated by the application. This process is analogous at the most to the use of an NCO to obtain an Object Reference for an object on a server, such that a proxy object therefor can be obtained (i.e., the well known use of an NCO).

Third, the passage at lines 38-59 of column 7 of *Hamilton* again focus on how an applet is downloaded to a client and used to interface with objects on a server. However, there is no discussion of naming services or of obtaining an Object Reference for an NCO.

Fourth, the passage at column 9, lines 55-65 of *Hamilton* discloses much of the same subject matter as the passage beginning at column 6, line 65 of *Hamilton*, only from the perspective of a server application. As above, the passage discloses that a client may access a network name server to obtain a machine address for a network machine. Other passages, e.g., column 9, lines 44-54, and column 9, lines 66 to column 10, line 4, disclose further specifics regarding network and object name servers with respect to accessing objects on a server. However, it is important to note that none of these passages disclose how a client obtains an Object Reference for an NCO.

Of note, *Hamilton* discloses both at column 7, line 10 and column 9, line 57, that a network client "may only be given the logical name for the network server" (where the "network server" refers to the server application within which an object resides). There is no disclosure or suggestion in the reference even that this logical name is downloaded to the client in response to

a request therefrom, much less that this name corresponds to an Object Reference for a Naming Context Object.

All that is disclosed in *Hamilton* is that a network name server may be accessed to obtain the machine address for a network server, and that an application program may be given the logical name for the network server, and use the network name server to resolve this to an address. However, the reference does not disclose how, once given a machine address, the object name server is accessed using this machine address. Moreover, the reference does not even disclose how the application program obtains the logical name for the network server in the first place. Thus, neither the access to the network name server, nor the access to the object name server, appears to correspond to Applicant's claimed concept of requesting and retrieving an Object Reference for an NCO.

The reference is, in effect, silent as to how an Object Reference for a naming service is obtained. *Hamilton* is therefore incapable of teaching to one of ordinary skill in the art, the concept of on-demand download of an Object Reference for a Naming Context Object.

As noted above, the Examiner also took Official Notice that a Naming Context Object is located on an object name server and is well known in the art. The fact that NCOs are known to be resident in object name servers, however, is still insufficient inasmuch as the reference still does not disclose that an Object Reference to this object is downloaded to the client in response to a request by that client.

As such, Applicant respectfully submits that *Hamilton* fails to disclose or suggest the on-demand download of an Object Reference for a Naming Context Object to a zero install client after installation of the zero install client on a client machine.

The Examiner additionally relies on *Kessler* for disclosing the specifics of a zero install client, and using an ORB naming service to obtain an Object Reference. In this regard, *Kessler* is identical to *Hamilton* in its disclosure of how a naming service is conventionally used in a CORBA environment. *Kessler* likewise fails to disclose or suggest how an Object Reference to a Naming Context Object is obtained.

In fact, it does not appear that the Examiner is relying on *Kessler* for anything more than the specifics of a zero install client. Regardless, Applicant has addressed in *Kessler* in the

previous Response, filed August 14, 2001. Specifically, Applicant has already pointed out how *Kessler*, at pages 15-16, does not disclose how an Object Reference for an NCO is obtained by a zero install client. Rather than rehash these arguments, Applicant refers the Board to this prior Response, where the lack of disclosure or suggestion in *Kessler* of the retrieval of an Object Reference for a Naming Context Object is discussed.

In order to render a claimed concept obvious, the references must clearly and unambiguously teach that concept to one of ordinary skill in the art. Applicant respectfully submits that the Examiner has failed to meet this burden, as neither *Hamilton* nor *Kessler* clearly discloses or suggests how an Object Reference for a Naming Context Object may be downloaded to a zero install client on demand. Accordingly, Applicant respectfully submits that the independent claims of Group I (claims 1, 7, 21 and 28), as well as the claims that depend therefrom (claims 2-6, 8-9, 11, 15-20, 22-27 and 29-32), are non-obvious over the prior art of record. Reversal of the Examiner's rejections of these claims are therefore respectfully requested.

B. The Group II Claims (claims 33-36 and 39) were improperly rejected under 35 U.S.C. § 103(a) as being unpatentable over *Hamilton* in view of *Kessler* and further in view of *Tang*.

Independent Group II claims 33 and 34 add the concept of a stringified Object Reference being used as the format for the Object Reference for the Naming Context Object. The Examiner relies on *Hamilton* and *Kessler*, with the addition of *Tang* to reject these claims. However, these claims still recite in various forms the on-demand download of an Object Reference for a Naming Context Object to a web browser or applet executing therein. *Hamilton* and *Kessler* do not disclose or suggest this claimed feature, as has been discussed above. Moreover, *Tang*, which is not even alleged to disclose this feature, and which is only relied upon to disclose the use of stringified Object References, likewise does not disclose or suggest this feature. Accordingly, the Group II claims are patentable over the prior art of record for the reasons discussed above. Reversal of the Examiner's rejections of the Group II claims are therefore respectfully requested.

IX. CONCLUSION

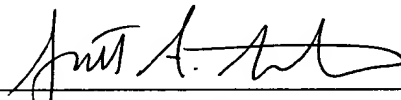
In conclusion, Applicant respectfully requests that the Board reverse the Examiner's rejections of claims 1-9, 11, 15-36 and 39, and that the Application be passed to issue. If there are any questions regarding the foregoing, please contact the undersigned at 513/241-2324. Moreover, if any other charges or credits are necessary to complete this communication, please apply them to Deposit Account 23-3000.

Respectfully submitted,

WOOD, HERRON & EVANS, L.L.P.

Date: 12 SEP 2002

2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
(513) 241-2324

By: 
Scott A. Stinebruner
Reg. No. 38,323

APPENDIX A: CLAIMS ON APPEAL (S/N 08/818,185)

1 1. (Once Amended) An apparatus comprising:
2 at least one processor;
3 a memory coupled to the at least one processor;
4 a computer program residing in the memory, said computer program enabling client
5 object - server object interaction for a client object located on a zero install client, said client
6 object - server object interaction being enabled by delivering an object reference for a naming
7 context object to said zero install client after said zero install client has contacted said computer
8 program.

1 2. (Once Amended) The apparatus of claim 1 wherein said computer program comprises a
2 web server.

1 3. (Once Amended) The apparatus of claim 1 wherein said contact with said computer
2 program is accomplished by a web browser located on said zero install client.

1 4. The apparatus of claim 2 wherein said object reference is stored in a web server directory.

1 5. The apparatus of claim 1 wherein said object reference comprises a stringified object
2 reference.

1 6. The apparatus of claim 1 wherein said object reference comprises a stringified object
2 reference and wherein said naming object comprises a root naming context object.

1 7. (Once Amended) A method for enabling client object - server object interaction , the
2 method comprising the steps of:
3 a) creating an object reference for at least one naming context object on a server system;
4 and

5 b) downloading said object reference from said server system to a client system after said
6 client system has contacted said server system, said client system being a zero install client.

1 8. The method of claim 7 wherein the step of creating an object reference comprises creating
2 a stringified object reference.

1 9. The method of claim 7 wherein said naming context object comprises a root naming
2 context object.

10. (Canceled)

1 11. (Once Amended) The method of claim 7 wherein said contact with said server system is
2 accomplished by a web browser executing on said client system and wherein the step of
3 downloading said object reference comprises downloading by a web server application.

12-14. (Canceled)

1 15. (Twice Amended) The method of claim 11 wherein said web browser comprises a Java-
2 enabled web browser.

1 16. (Twice Amended) The method of claim 11 wherein said web browser comprises a Java-
2 enabled web browser containing a CORBA compliant Java Object Request Broker.

1 17. (Twice Amended) The method of claim 11 wherein said server system further includes a
2 location service application, and wherein said step of downloading said object reference is
3 performed by a web server application in said server system.

1 18. (Once Amended) The method of claim 17 wherein said web server includes a name
2 object server.

1 19. (Once Amended) The method of claim 17 further comprising the steps of:
2 downloading an applet from said web server to said web browser and running said applet
3 on said web browser;
4 downloading an object request broker from said web server;
5 requesting said object reference from said web server; and
6 retrieving a proxy for said naming context object using said retrieved object reference.

1 20. The method of claim 19 wherein the step of downloading said object request broker
2 comprises downloading the class of said object request broker from the web server.

1 21. (Once Amended) A program product comprising:
2 (A) a computer program, said computer program enabling client object - server object
3 interaction for a client object located on a zero install client, said client object - server object
4 interaction being enabled by delivering an object reference for a naming context object to said
5 zero install client after said zero install client has contacted said computer program; and
6 (B) signal bearing media bearing said computer program.

1 22. The program product of claim 21 wherein the signal bearing media comprises recordable
2 media.

1 23. The program product of claim 21 wherein the signal bearing media comprises
2 transmission media.

1 24. (Once Amended) The program product of claim 21 wherein said naming context object
2 is delivered to a remote web browser located on said zero install client.

1 25. (Once Amended) The program product of claim 21 wherein said computer program
2 comprises a web server having access to said object reference.

1 26. The program product of claim 21 wherein said object reference comprises a stringified
2 object reference.

1 27. The program product of claim 21 wherein said object reference comprises a stringified
2 object reference of a root naming context object.

1 28. (Once Amended) A program product comprising:

2 (A) an applet, said applet being used to retrieve an object reference for a naming context
3 object from a server apparatus; and

4 (B) signal bearing media bearing said applet.

1 29. The program product of claim 28 wherein the signal bearing media comprises recordable
2 media.

1 30. The program product of claim 28 wherein the signal bearing media comprises
2 transmission media.

1 31. The program product of claim 28 wherein the signal bearing media comprises the
2 Internet.

1 32. The program product of claim 28 wherein said server apparatus comprises a web server
2 having access to said object reference.

1 33. An apparatus comprising:

2 at least one processor;

3 a memory coupled to the at least one processor;

4 a server system, said server system comprising:

5 a) at least one object server, said at least one object server including a naming
6 context object;

b) a web server, said web server having access to a stringified object reference for said naming context object, wherein said web server downloads said stringified object reference to a web browser when said stringified object reference is requested by said web browser.

34. (Once Amended) A method for enabling client object - server object interaction, the method comprising the steps of:

- a) creating an object reference for at least one naming context object;
- b) storing said object reference in a web server directory;
- c) downloading an applet from a web server to a web browser and running said applet on said web browser;
- d) downloading an object request broker from said web server;
- e) requesting said object reference from said web server; and
- f) retrieving a proxy for said naming context object using said retrieved object reference.

35. The method of claim 34 wherein the object request broker comprises a Java object request broker.

36. The method of claim 34 wherein the object reference comprises a stringified object reference.

37-38. (Canceled)

39. (Once Amended) The method of claim 34 wherein the step of downloading an object request broker comprises downloading a Java object request broker.

PATENT

IBM/161
Confirmation No. 9188

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Randall R. Schnier
Serial No.: 08/818,185
Filed: March 14, 1997
For: A BOOTSTRAPPING TECHNIQUE FOR DISTRIBUTED OBJECT CLIENT SYSTEMS

Art Unit: 2152
Examiner: Thong Vu
Atty. Docket No.: IBM/161

TRANSMITTAL

Assistant Commissioner for Patents
ATTENTION: Board of Patent Appeals and Interferences
Washington, DC 20231

RECEIVED

SEP 25 2002

Sir:

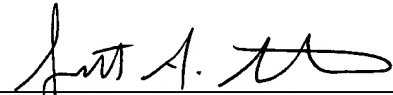
Technology Center 2100

We are transmitting herewith the attached:

Transmittal (in duplicate) containing Certificate of Mailing Under 37 CFR 1.8
Communication
Revised Appeal Brief (in triplicate)
Reply Postcard

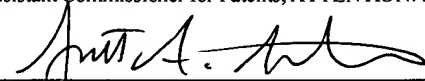
Please charge any additional fees or credit overpayment to Deposit Account No. 23-3000.
A duplicate of this sheet is enclosed.

WOOD, HERRON & EVANS, L.L.P.
2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
(513) 241-2324

By: 
Scott A. Stinebruner Reg. No. 38,323

CERTIFICATE OF MAILING 37 CFR 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Assistant Commissioner for Patents, ATTENTION: Board of Patent Appeals and Interferences, Washington, D.C. 20231 on September 12, 2002.


Scott A. Stinebruner Reg. No. 38,323

BOARD OF PATENT APPEALS AND INTERFERENCES

SEP 23 PM 4:40

RECEIVED